LA-UR-21-30687
Approved for public release; distribution is unlimited.

Title: Release of the NDI 2.2.0

Author(s): Saller, Thomas

Intended for: Report

Issued: 2021-10-27

Los Alamos
NATIONAL LABORATORY
EST. 1943

To/MS: Distribution
From/MS: Thomas G. Saller, CCS-2, MS B265
Phone/Cell: 7-8206/551-2077
Symbol: CCS-2-21-032
Date: October 27, 2021

memorandum

Computational Physics Group
CCS-2, Mail Stop D413
Phone: 505-667-7029 Fax: 505-665-4972

Release of the NDI 2.2.0

Executive Summary

The Nuclear Data Interface 2.2.0 has been released. It includes the addition of charged particle dE/dx (stopping power) data, the ability to read in NDI-formatted binary data, corrections to TN data, and other minor changes and bug fixes.

1. Introduction

The Nuclear Data Interface (NDI) 2.2.0 has been released. It has four major components. First, it adds charged particle data as per the recent L2 milestone [1]. This includes stopping powers (dE/dx) in a new NDI data format. The stopping power capability has currently been adopted by LAP (Lumos) and will be added to EAP and Jayenne in the future.

Second, NDI 2.2.0 can now read in NDI-formatted binary multigroup data. There is a script (convert.c) that will take ASCII NDI data and convert it to binary, which NDI can then read in. Preliminary tests have shown that this can reduce NDI read times by more than a factor of four (with large variance).

Third, NDI 2.2.0 has corrected TN data, as per CJ Solomon's work [2]. Problems with energy balances were found with the lanl01, lan04, and lanl04e libraries. These have been corrected to preserve energy balance, and the new data is given as lanl01_2020, lanl20, and lanl20e, respectively.

Finally, there were several other minor bug fixes or feature additions. The build system was fixed and modified to always use "-fp-model -precise" for Intel Release builds. Warning and error numbers were changed and a warning was added for calls to `ndi_get_size*` with a gendir handle when the sought after isotope is missing in the gendir.all for the specified library. The gendir was fixed for the hr_118 library, and the maximum string length for things like gendir paths was increased.

2. Release Summary

The following summarizes the changes from NDI 2.1.4 to 2.2.0:

- Added the ability to read in charged particle stopping powers (dE/dx).
- Added the ability to read in NDI-formatted binary data.
- Fixed the gendir.all:
  - For 94240 in library hr_118, the s0 values were missing the E for their exponential (e.g., 6.00+04 instead of 6.00E+04). The E has been added in.

- Modified fmangle.h to compile with XL.
- Changed several warnings and errors and added error strings. Most importantly, changed warning +54 to error -34 (library not in gendir).
- Update lanl01, lanl04, and lanl04e TN libraries to conserve energy balance [2].
- Increased maximum string lengths (e.g., for gendir paths) from 133 to 512 characters.
- Changed the build system to allow for a build directory to have any name, not just "build" (work performed by CJ Solomon).
- The location of the binary cross section file created for the NDI binary test has been corrected.
- "-fp-model -precise" was added for Intel Release builds to eliminate a small difference during group collapses between Release and Debug. A new test was added to cover this problem.
- The warning 19 was added for calls to `ndi_get_size*` with a gendir handle when the sought after isotope is missing in the gendir for the specified library. This warning, "Isotope not found in gendir" is the warning equivalent of error -8.
- The version has been updated to 2.2.0.

3. **Installation on HPC platforms**

Table 1 gives the installation locations for each HPC machine and the compiler used. For each compiler, the default was chosen based on `module load`. For instance, on Snow `module load gcc` yielded GCC 9.3.0. Installations with other compilers will be provided upon request.

For each installation, both release (`libndi.a`) and debug (`libndi-debug.a`) versions are available. `ndicl` and `ndicl.py` are available on Snow in `/usr/projects/data/nuclear/ndi/2.2.0/x86_64-linux/bin`

TABLE 1: NDI 2.2.0 Installations

Machine	**Install Location**	**Compiler**
Snow/Fire/Ice	`/usr/projects/data/nuclear/ndi/2.2.0/x86_64-linux/`	GCC 9.3.0
Snow/Fire/Ice	`/usr/projects/data/nuclear/ndi/2.2.0/x86_64-linux_intel/`	Intel 19.1.3
Trinitite	`/usr/projects/data/nuclear/ndi/2.2.0/x86_64-linux_tt/`	Intel 19.1.3
Trinity	`/usr/projects/data/nuclear/ndi/2.2.0/x86_64-linux_tr/`	Intel 19.1.3
Capulin/Thunder	`/usr/projects/data/nuclear/ndi/2.2.0/arm-gnu/`	GCC 7.5.0
Darwin (Power 9)	`/usr/projects/data/nuclear/ndi/2.2.0/ppc-gnu/`	GCC 4.8.5
Darwin (Power 9)	`/usr/projects/data/nuclear/ndi/2.2.0/ppc-xl/`	XL 16.1.1.7
Darwin	`/usr/projects/data/nuclear/ndi/2.2.0/x86_64-linux/`	GCC 4.8.5
Sierra/RZansel	`/usr/gapps/lanl-data/nuclear/ndi/2.2.0/ppc-gnu/`	GCC 4.9.3
Sierra/RZansel	`/usr/gapps/lanl-data/nuclear/ndi/2.2.0/ppc-xl/`	XL 16.1.1.7

Data is in the `2.2.0/share/` directory, and release notes are in `2.2.0/docs/`.

4. **Corrected Thermonuclear Data**

In [2], Solomon discovered that several TN reactions in the `lan01`, `lanl04`, and `lanl04e` libraries failed to conserve energy, some to more than 20% errors. Solomon discovered three problems with these libraries:

1. imbalances caused by omission of the $\frac{3}{2}k_BT$ factor from the $\overline{E}_{in}$ in the laboratory frame;
2. imbalances in the three-body breakup reactions, e.g., d + ^{6}Li $\rightarrow$ p + t + α
3. imbalances in endoergic (negative Q-value) reactions.

Adding the missing $\frac{3}{2}k_BT$ factor improved many of these reactions. For the others, Solomon developed a phenomenological correction to fix the other imbalances. This correction was applied to all reactions for consistency, with negligible effect on reactions that already conserved energy. For more on these corrections, see [2].

The new libraries, their ZAID extensions, and their corresponding original libraries are:

- `lanl01_2020` (.012ztn) from `lanl01`,
- `lanl20` (.042ztn) from `lanl04`,
- and `lanl20e` (.043ztn) from `lanl04e`.

These new libraries are in the `share/tn` directory.

5. **Error and Warning Changes**

Several warnings and changes were modified, and error strings were added. A warning +19 was added for calls to `ndi_get_size*` with a gendir handle when the sought after isotope is missing in the gendir for the specified library. It is equivalent to the error -8. (Note: this change may require modification of host codes.) These changes are given in Table 2, and the new error strings are now available through `ndi2_get_string_val_n`.

TABLE 2: Warning and Error changes

Current #	**Old #**	**String**
+16	+16	"Invalid interpolation type."
+17	+30	"Quantity outside maximum and minimum library values."
+18	+77	"New group structure is default but old group structure is not."
+19	N/A	"Isotope not found in gendir."
-32	-32	"Reaction product not in depletion set."
-33	-33	"Last index greater than number of particles."
-34	+54	"Library not found in gendir."
-35	-99	"Invalid particle handle."
-36	-999	"Bad particle."

6. **Stopping Powers (dE/dx)**

As part of the FY 2020 charged particle L2 milestone [1] stopping powers were added to NDI 2.2.0. The following, largely taken from [1], describes these changes in detail.

6.1. **Gendir**

A new charged particle section, `dedx`, has been added to the gendir.all file. The location of all charged particle data should be referenced in this section. NDI currently assumes that the charged particle data available in a gendir entry is the same as the multigroup neutron gendir data (e.g., zaid is "z", library is "l", and atomic weight is "aw"). This is consistent with other NDI data types.

6.2. **The New Library and Particle Types**

A new library type, 'x', has been added to NDI to refer to dE/dx data. Particle types are assumed to be equal to the ZAID of the charged particle (e.g., 1001 for H-1 and 2004 for He-4). There are currently three libraries: `rpa_cut`, `rpa_nocut`, and `bps_nocut` (these were previously named `standard`, `nocut`, and `bps`). RPA and BPS refer to the model used to generated the data (RPA = random phase approximation and BPS = Brown, Preston, and Singleton). Cut and nocut refer to whether or not the data includes large-angle cutoff (`cut`) or not (`nocut`). A cutoff is typically used for Monte Carlo transport.

6.3. **Charged Particle Structure**

A new struct has been defined for charged particle data. "dedx_status" includes the following data:

- `dedx_zaid zaid`
- `int set_flag`
- `int num_grps`
- `int num_densities`
- `int num_temps`
- `int num_targets`
- `int *targets`
- `char info[MAX_IN_STR_LEN]`
- `char date_processed[MAX_IN_STR_LEN]`
- `char date_source[MAX_IN_STR_LEN]`
- `char library_name[MAX_IN_STR_LEN]`
- `double at_wgt`
- `double awr`
- `stat_flag energies`
- `stat_flag densities`
- `stat_flag temps`
- `stat_flag free_e_dedx`
- `multi_stat target_dedx`

A `dedx_status` struct, `dedx_stat` has been added the the `state_struct` struct.

6.4. **Keywords and Commands**

The following presents keyword-function call pairs that are valid with the dedx library. The term "points" refers to a point on the energy-density-temperature grid that corresponds to the dE/dx data. More may be added as requested.

- `ndi2_get_int_val`
  - NDI_NUM_GRPS – returns the number of energy points.
  - NDI_NUM_DENSITIES – returns the number of density points.
  - NDI_NUM_TEMPS – returns the number of temperature points.
  - NDI_NUM_TARGET – returns the number of target ions.
- `ndi2_get_int_val_n`
  - NDI_TARGET_ZAID – returns the ZAID of target $n = [0, \text{num\_targets}-1]$.
- `ndi2_get_int_vec`
  - NDI_TARGET_ZAID – returns an array of target ZAIDs (as integers).
- `ndi2_get_float_val`
  - NDI_AT_WGT – returns the atomic weight of the incident charged particle.
  - NDI_AWR – returns the atomic weight ratio of the incident charged particle.
- `ndi2_get_float_vec`
  - NDI_ENERGIES – returns the energy grid for the dE/dx data.
  - NDI_DENSITIES – returns the density grid for the dE/dx data.
  - NDI_TEMPS – returns the temperature grid for the dE/dx data.
  - NDI_DEDX – returns the free-electron dE/dx data.
- `ndi2_get_float_vec_x`
  - NDI_TARGET_DEDX – returns the dE/dx data for target x, where x is a ZAID (as a string).
- `ndi2_get_string_val`
  - NDI_ZAID – returns incident particle's ZAID.
  - NDI_INFO – returns information on the data file.
  - NDI_DATE_PROCESSED – returns the date the data was processed.
  - NDI_DATE_SOURCE – returns the date the data was sourced.
  - NDI_LIBRARY_NAME – returns the library name.
- `ndi2_get_size`
  - NDI_ZAID – returns the size of the ZAID string.
  - NDI_INFO – returns size of the information string.

  - NDI_DATE_PROCESSED – returns the size of the string containing the data was processed.
  - NDI_DATE_SOURCE – returns the size of the string containing the data was sourced.
  - NDI_LIBRARY_NAME – returns the size of the library name string.
  - NDI_NUM_TEMPS; NDI_NUM_GRPS; NDI_NUM_DENSITIES; NDI_NUM_TARGETS – all return 1.
  - NDI_ENERGIES – returns the number of energies.
  - NDI_DENSITIES – returns the number of densities.
  - NDI_TEMPS – returns the number of temperatures.
  - NDI_TARGET – returns the number of targets.
  - NDI_DEDX; NDI_TARGET_DEDX – returns the number of stopping power values, which is the number energies times the number of densities times the number of temperatures.
- `ndi2_get_size_n`
  - NDI_TARGET_ZAID – returns 1 (because the target ZAID is stored as an integer).
  - NDI_TARGET_DEDX – returns the number of stopping powers for the target, which is the number of energies times the number of densities times the number of temperatures.
- `ndi2_get_size_x`
  - NDI_TARGET_ZAID – returns 1 (because the target ZAID is stored as an integer).
  - NDI_TARGET_DEDX – returns the number of energies times the number of densities times the number of temperatures.

6.5. **dE/dx Data**

The original dE/dx (stopping power) data is found at

`/usr/projects/data/nuclear/mc/dedx/all/`

This folder contains tabulated data produced by the FermiASC$_{\text{NEW}}$ code.

The equivalent NDI data is at

`/usr/projects/data/nuclear/ndi/2.2.0/share/cp/`

with `rpa_cut`, `rpa_nocut`, and `bps_nocut` libraries. Each library has a unique ZAID extension: `.000dx` for `rpa_cut`, `.001dx` for `rpa_nocut`, and `.002dx` for `bps_nocut`.

Two python scripts have been written to convert ACE-format dE/dx data to the NDI format. `convert.py` is the main script, and it uses the function `write_data` function from `write_data.py`.

6.6. **Format**

dE/dx data is given in a new NDI format. Like all other NDI data formats, it contains a series of keywords followed by data pertaining to that keyword. The data is given in log-form (like the original ACE data). To get the true energy, density, temperature, etc., one must take the exponential of the data (i.e., e^{data}). A

sample data file is presented below:

```
dE/dx information translated from ACE to NDI
zaid
    1001.000dx
info
    This data has been converted from ACE to NDI
library_name
    rpa_cut
date_source
    unknown
date_processed
    2020-04-16
awr
    0.998626981306
at_wgt
    1.007280E+00
num_grps
    91
num_dens
    28
num_temps
    28
energies
    -8.517193e+00 -8.363688e+00 -8.210182e+00 -8.056676e+00 ... (91 total entries)
densities
    4.835429e+01 4.903653e+01 4.971878e+01 5.040103e+01 ... (28 total entries)
temperatures
    -1.151293e+01 -1.083068e+01 -1.014843e+01 -9.466183e+00 ... (28 total entries)
free_e_dedx
    -4.002136e+01 -4.012272e+01 -4.022826e+01 ... (91*28*28 total entries)
num_targets
    15
target_ion_dedx
    1001
    -4.861460e+01 -4.864614e+01 -4.869095e+01 ... (91*28*28 total entries)
    1002
    -4.908976e+01 -4.914400e+01 -4.920720e+01 ... (91*28*28 total entries)
    ... (15 total target ion dE/dx sets of data, each 91*28*28 total entries of data)
end
```

Here the ZAID of the incident particle is `1001.000dx` (H-1). The library is "rpa_cut," based on the library_name keyword and the ZAID extension `.000dx`. It has 91 energy points, 28 temperatures, 28 densities, and 15 target ions. For each target ion, there is a ZAID and 91x28x28 dE/dx values.

The stopping power data is given in this order (showing energy/density/temperature points), where G is

the total number of energies, N is the total number of densities, and M is the total number of temperatures:

```
ZAID
(E1, D1, T1) (E2, D1, T1) (E3, D1, T1) ... (EG, D1, T1)
(E1, D2, T1) (E2, D2, T1) (E3, D2, T1) ... (EG, D2, T1)
...
(E1, DN, T1) (E2, DN, T1) (E3, DN, T1) ... (EG, DN, T1)
(E1, D1, T2) (E2, D1, T2) (E3, D1, T2) ... (EG, D1, T2)
(E1, D2, T2) (E2, D2, T2) (E3, D2, T2) ... (EG, D2, T2)
...
(E1, DN, TM) (E2, DN, TM) (E3, DN, TM) ... (EG, DN, TM)
```

6.7. **Comparisons**

The following plots (Figs. 1 to 4) compare dE/dx data from NDI to the original tabular data from the FermiASC$_{\text{NEW}}$ code[1]. Each run tracks a single, fully ionized 3.5 MeV alpha particle through an infinite medium of 10 g/cc 50-50 DT gas at 8 keV (Ti = Te = Tr = 8 keV). In this particular problem, hydrodynamics and coupling between the particles and background plasma were both disabled. This allows the particle to be tracked as it slows down without heating the plasma, so the plasma stays at a constant density and temperature for the entire duration of the run. These results are from before the change in library name. As a reminder, `standard` is now `rpa_cut`, `nocut` is `rpa_nocut`, and `bps` is `bps_nocut`.

There are four plots. Two (Figs. 1 and 2) show dE/dx as a function of time and as a function of space, respectively, while the third and fourth (Figs. 3 and 4) show energy as a function of time and of space. Both dE/dx plots show the same general behavior – a long sloped region leading ending in a sharp increase just before the particle is thermalized. This trend is expected; the region where the stopping power slowly decreases is the electron dominated region and the sharp increase is due to the particle-ion interaction. The narrower peak in Fig. 2 occurs because the charged particle is moving slowly towards the end. Most importantly, in each case the relative difference between NDI and the reference was $< 10^{-10}$. Similar agreement is seen with the energy plots, in which the charged particle gradually loses energy.

[1]Plots and explanations provided by Edward Norris of XCP-1

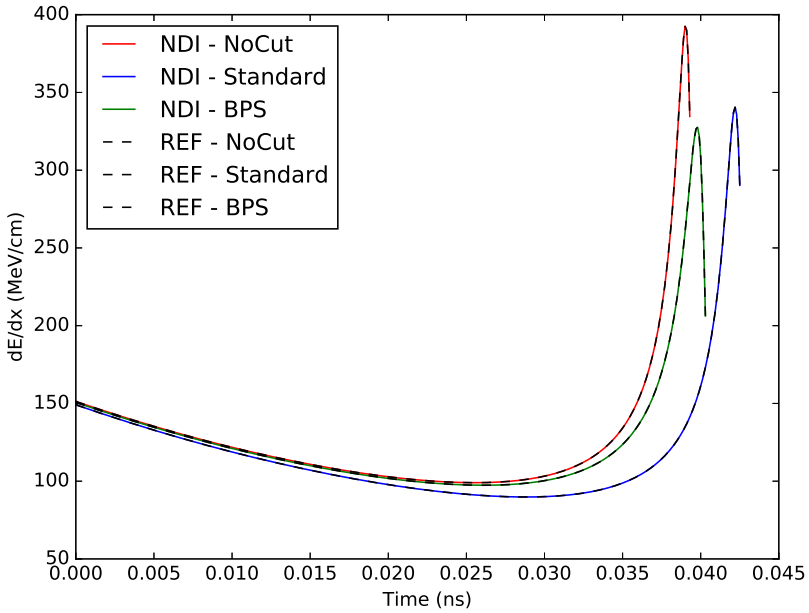

FIG. 1: dE/dx as a function of time

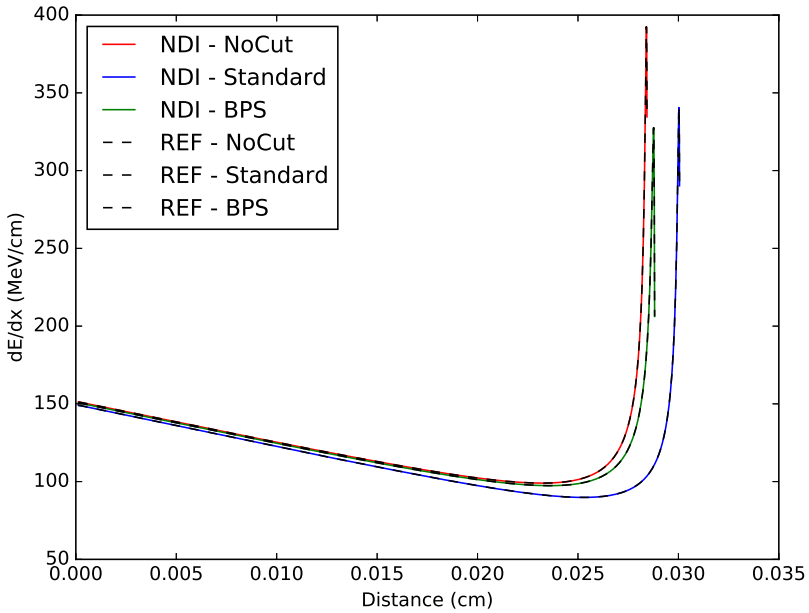

FIG. 2: dE/dx as a function of space

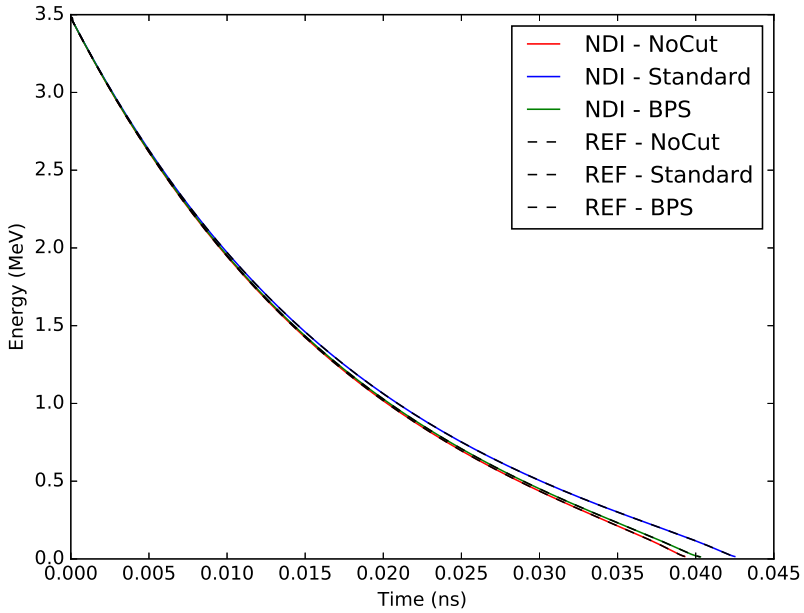

FIG. 3: Energy as a function of time

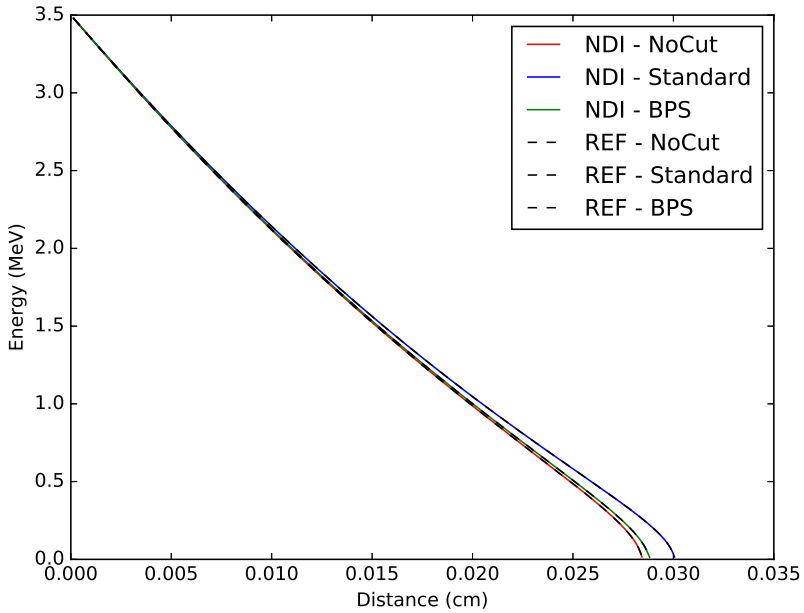

FIG. 4: Energy as a function of space

7. **Binary data**

One major complaint about NDI 2 is that it is slow, particularly reading in multigroup neutron data. Because 618 energy groups is the default, each neutron data file is large (order 10 MB). Typically tens of isotopes are used in a run, often at 15 or 23 temperatures. Reading in all of this data can take tens of minutes. If one could speed this up, it would have a significant effect on input processing, which would be particularly helpful for when someone wants to run thousands of simulations.

NDI 2.2.0 adds the ability to read in binary multigroup neutron and photon data. While speedups are very problem dependent, current testing has shown speed-ups of 3x to 6x, which correspond to a 1.7x to 3.3x speedup in input processing in PARTISN [3].

A C program has been written to convert ASCII multigroup data files to binary ones. So far only the MTMG01 multi-temperature library has been converted (found in `share/sn/mtmg01_binary` with its own gendir, `gendir.bin`), but others can be as requested. This data is currently only available on the CTS machines.

This has only been implemented for multigroup neutron and photon data. Thermonuclear data is typically only a single file of 3 to 10 MB that includes all pertinent reactions. Production/depletion data is less than 1 MB, while radiochemistry data can be up to 5MB. Both, however, also only include a single data file. Therefore, multigroup neutron/photon data remains the most significant source of long runtimes. Other data types may be converted to binary in the future.

7.1. **Code changes**

NDI's ability to read binary data files is highly dependent on how the data files are written. The conversion program uses `fwrite` to write to binary data files with set types. For example, floats are always written as doubles and strings as 512 characters (the NDI default).

Because `fwrite` was used to generate the binary files, `fread` is used to read them. Like with `fscanf`, `fread` needs to know the data type it is reading in. This greatly simplified some uses of `fgets`. `fgets` grabs a line of character, which must then be parsed. NDI's new binary data files have no lines, just individual bits of data read in with `fread`.

7.2. **Gendir changes**

NDI determines whether its data is binary or ASCII based on the gendir file. The gendir file has always contained a file type indicator, `ft`. Previously, `ft` was set to `asc` for ASCII data file, and NDI would return an error for any other file type. `ft` can now be set to `bin` to indicate binary data.

Naming the modified data files and their full ZAIDs is still an open question. A separate gendir for the binary data is currently required, but that is only because of the current ZAID designations. For NDI 2.2.0, the ZAIDs are identical to the original ones with the addition of "b" at the end. For example, `1001.121nm` became `1001.121nmb`. Data file names are also the same as before with the addition of a "b." Library names are currently unchanged.

The extra comment lines at the beginning of the ASCII data files have been removed. Now any informa-

tion the data creator may want to convey to the user should be placed in the information section.

A binary-specific gendir has been added in the `share` directory as `share/gendir.bin`.

7.3. **Timing results**

To test NDI's new binary data feature, the MTMG01 library was converted to binary. Tests were run with different numbers of isotopes with both one temperature and multiple temperatures (15). Three timings from PARTISN's input module were analyzed: the time spent reading in NDI data, the time spent in Block 4 (which involves reading in data from the gendir file), and the total time spent processing the inputs.

Timings are given in Tables 3 and 4. Each table shows the improvement from reading in ASCII data to reading in binary data as a ratio of the ASCII timing to binary timing. Table 3 covers single temperature data, while Table 4 shows multi-temperature results. The number of isotopes read in range from 1 to 30. Each case included 5 fissile isotopes (all uranium), except for the single isotope case.

TABLE 3: Single Temperature Improvement from ASCII to Binary

# Isotopes	# Fissile Isotopes	Block 4	Data Reading	Total
1	1	1.090534979	5.450819672	1.737654321
5	5	1.197265625	5.425249169	2.517911975
10	5	1.186335404	4.625711035	2.259430605
20	5	1.155731795	3.739009072	1.977688532
30	5	1.145695364	3.317907445	1.872900889

From Table 3, we see a few things of note.

1. As expected, reading in binary data is faster than reading in ASCII data.
2. Data reading speedups are between 3x and 6x, a significant improvement.
3. Block 4 timings are slightly improved. This is because most of Block 4's NDI calls are to the gendir file, which has not changed in format or length.
4. Overall input timing is improved, but not as significantly. Reading in data is only a portion of the overall time spent in the input module, especially when dealing with a single temperature.

While these results were encouraging, of more interest is the speedup for multi-temperature cases. Table 4 shows the improvement for multi-temperature cases with 15 temperatures.

TABLE 4: Multi-temperature Improvement from ASCII to Binary

# Isotopes	# Fissile Isotopes	Block 4	Data Reading	Total
1	1	1.383561644	5.892658509	2.884329744
5	5	1.400543901	5.839799331	3.37904147
10	5	1.347266539	5.129179679	2.892282461
20	5	1.320494104	4.54482468	2.545706861
30	5	1.293455607	4.182491333	2.36576443

Some important notes from the multi-temperature runs:

1. Greater speedups were seen for multi-temperature runs than the single temperature run. Before, the smallest speedup for reading in NDI data was 3.3x, and the max was 5.4x. For multi-temperature, the minimum was 4.1x and the maximum was 5.9x.
2. Block 4 timings improved more than with a single temperature, but still not significantly.
3. This increase meant that total input times improved by 2.3x to 3.3x, rather than 1.7x to 2.5x for a single temperature.

From this it appears that we can expect a speedup of ≈5x reading in NDI data, and an overall speedup of 2x or greater for PARTISN input processing.

8. **CP2011 and CP2020 Neutron Data**

NDI 2.2.0alpha, 2.2.0beta, and 2.2.0beta-1 included neutron data that included the production of charged particles. The libraries were named `cp2011` and `cp2020` based on when they were created. Problems were discovered with KERMA balances, and so this data has been removed in the final release 2.2.0. This data will be included in a future version of NDI when they have been fixed.

9. **Bug Fixes and Miscellaneous Upgrades**

Finally, NDI 2.2.0 contains a few bug fixes and additional minor features. First, the gendir.all file had a typo for Pu-240 (94240) for the hr_118 library. The gendir entries for each lacked the exponential E for their background cross section (s0) values. For example, 94240.443nm had `s0=6.00+04`. This was corrected to `s0=6.00E+04`, and the others were corrected similarly.

Second, there was a problem with compiling NDI with the XL compiler. This was corrected by including `#define NDI_NO_NAME_SUB` in fmangle.h and ndi.h (this work was performed by Paul Henning).

Third, the maximum string length for things like gendir paths was increased from 133 to 512 characters. This was to accommodate longer paths potentially seen when using SPACK.

Fourth, a gendir checking script (check_gendir.py) was added to test gendir files for potential bugs. This includes a case in which the next gendir entry was placed before the `end` from the previous entry, which led to NDI reading in a blank ZAID. The blank ZAID led to a NULL string that caused NDI to crash when attempting to call `strlen` on the NULL string. In addition to the new checking script, NDI has been modified to handle such an error.

Fifth, NDI now builds with position independent code (-fPIC) by default.

Sixth, the build system was also modified by Solomon to allow for the build directory to have any name (not just "build").

Finally, Intel Release also builds with "-fp-model -precise" by default to eliminate a small difference between Release and Debug during group collapses. A new test was added to cover this problem.

References

[1] M. Paris et al., "Charged particle transport FY2020 Level-2 Milestone – Final Report," Internal Report, August 2020.

[2] C.J. Solomon, "Energy Balance Discrepancies in the `lanl04e` TN Data," Internal Memo, XCP-2:20-005, August 2020.

[3] R. E. Alcouffe et al., "PARTISN: A Time-Dependent, Parallel Neutral Particle Transport Code System," Los Alamos National Laboratory report LA-UR-17-29704 (Revised September 2020).

Distribution:

Group Office, ccs2go@lanl.gov
Patrick Talou, talou@lanl.gov
Jeremy Conlin, jlconlin@lanl.gov
Kent Parsons, dkp@lanl.gov
NDI Users' Group, ndi_ug@lanl.gov